M&G No. 60001.0261USI1/MS 303914.1.1

MECHANISM FOR DOWNLOADING SOFTWARE COMPONENTS FROM A REMOTE SOURCE FOR USE BY A LOCAL SOFTWARE APPLICATION

Related Applications

This patent application is a continuation-in-part under 35 U.S.C. §120 of United States Patent Application No. 10/164,260 filed on June 5, 2002, which is incorporated herein by reference.

Technical Field

This invention relates to a mechanism for downloading software components from a remote source for use by a software application.

Background of the Invention

Computer software applications allow users to create a variety of documents to assist them in work, education, and leisure. For example, popular word processing applications allow users to create letters, articles, books, memoranda, and the like. Spreadsheet applications allow users to store, manipulate, print, and display a variety of alpha-numeric data. Such applications have a number of well-known strengths, including rich editing, formatting, printing, calculation, and offline editing.

With the advent of such useful software applications, it has become popular to develop and use a variety of intelligent documents where such documents may point to and download via a software application helpful solutions that enhance the performance of such documents. For example, a resume document may be developed as a template with open fields for each section of the document such as the "experience" section or the "education" section. The document may point to a solution that provides helpful functionality. For example, after downloading a particular solution, a user of the document may be provided a tutorial on completing resumes, or the user may be

provided helpful information as the user focuses an insertion point on certain fields of the document. For example, if the user focuses on the education section of the document, the user may be provided helpful information as to how many education entries are suggested.

Unfortunately, such systems typically provide a user with a static "one-size-fits-all" solution that is the same solution regardless of the user's profile including the user's identity, job description, location, or other information associated with the user that may affect the usefulness of a particular solution. Accordingly, there is a need for enabling users to download document solutions that are dynamically configured based on the user's individual profile.

Because of the growth of available document solutions, large corporate, educational, civic, government, and other organizations may desire to make great numbers of document solutions available to their users. Unfortunately, associating great numbers of document solutions to individual user computers or software applications is inefficient, requires excessive computer memory, and requires frequent updating. Accordingly, there is a need for a collection of document solutions that contains information necessary for users to obtain desired or required document solution components.

It is with respect to these and other considerations that the present invention has been made.

Summary of the Invention

The present invention provides methods and systems for dynamically configuring document solutions and for obtaining desired or required components of such document solutions for use with computer-generated documents. According to one aspect of the invention, if a document is associated with a document solution, a query is made to a local namespace (schema) library to determine whether the document solution has already been downloaded to the user's computer or is otherwise available remotely for use in providing the desired solution to the document. If the solution is not presently available, or if the solution requires updating, the document

may point to a manifest of document solutions. If the document points to a manifest of document solutions, a request for the components of the desired solution is sent to the manifest. Along with the request, information about the user is also passed to the manifest. According to one aspect of the invention, the manifest is configured as an active server page. At the manifest, identification information for the user is utilized to query a user database to determine a profile of the user such as the user's job description, location, security clearance, and the like. Based on the user's profile, the document solution is configured dynamically to include solution components most helpful to this particular user. A configured solution is passed back to the user for use with the user's document.

According to another aspect of the invention, a manifest collection is provided for serving as a repository of document solution location information for many different namespaces (schemas) that may be associated with one or more user documents. Rather than downloading great numbers of document solutions to a user's computer, the user's computer is configured to point to a uniform resource locator associated with a manifest collection. When a user document references a particular namespace, such as an Extensible Markup Language (XML) namespace associated with an XML markup solution applied to the document, the user's document queries the manifest collection for a referenced solution. At the manifest collection, one or more namespaces associated with the user's document are used to locate a solution identification and location. Once the identification and location of a particular solution is obtained, components of that solution may be obtained from one or more remotely maintained solution manifests as described above.

These and other features, advantages, and aspects of the present invention may be more clearly understood and appreciated from a review of the following detailed descriptions of the disclosed embodiments and by reference to the appended drawings and claims.

Brief Description of the Drawings

Fig. 1 is a simplified block diagram illustrating a client-server architecture for use in conjunction with an embodiment of the present invention.

Fig. 2 is a block diagram of a computing system and associated peripherals and network devices that provide an exemplary operating environment for the present invention.

Fig. 3 is a computer screen display of a software application graphical user interface through which is displayed a document and associated contextually sensitive actions and help content according to an embodiment of the present invention.

Fig. 4 is a flow chart illustrating a method for downloading components required for operating and for providing functionality to a client-side software application.

Fig. 5 illustrates a computer-generated dialog box for offering multiple document solutions to the user.

Fig. 6 illustrates a computer-generated dialog box for assisting a user with downloading components required by a software application.

Fig. 7 illustrates a computer-generated dialog box for prompting a user of the need to connect to the internet for downloading components according to an embodiment of the present invention.

Fig. 8 illustrates a computer-generated dialog box for warning a user of a potential security risk associated with downloading components to the user's computer.

Fig. 9 illustrates a computer-generated dialog box for alerting a user of a problem associated with downloading components required by the user's software application.

Fig. 10 is a block diagram illustrating a system architecture for use in conjunction with an embodiment of the present invention showing interaction between a user's computer and a remotely maintained manifest and manifest collection.

Fig. 11 is a flowchart illustrating a method for obtaining a dynamically configured document solution for use in conjunction with a user document according to embodiments of the present invention.

Detailed Description

Referring now to the drawings in which like numerals refer to like parts or components through the several Figures. Fig. 1 is a simplified block diagram illustrating a client-server architecture for use in conjunction with an embodiment of the present invention. All components and files necessary to operate or fully implement the functionality or solutions available to a software application 110 are identified and are assembled on a local library of software components in the schema library 105. For a detailed description of the use of an operation of Namespace or schema libraries, see U.S. Patent Application entitled "System and Method for Providing Namespace Related Information," U.S. Serial No. 10/184,190, filed June 27, 2002, and U.S. Patent Application entitled "System and Method for Obtaining and Using Namespace Related Information for Opening XML Documents," U.S. Serial No. 10/185,940, filed June 27, 2002, both U.S. patent applications of which are incorporated herein by reference as if fully set out herein.

All components and files that may be utilized to update or add to functionality available to the application 100 are identified and are assembled on a manifest 38 which may be located in a remote library of software components on a remote web server 49 accessible by the user via the user's computer 20. If the user is informed that the components on her client-side computer 20 should be updated, or that corrections or improvements to existing components are available, or that new functionality is available that will transform the user's existing application 100 and document 110 into a "smart" application and "smart" document, the user may connect to the web server 49 via the Internet 51, or via an intranet 52, to download the required components. Alternatively, the user may connect to the web server 49 via any suitable distributed computing environment, including wireline and wireless telecommunications networks, for allowing the user to connect to the web server 49 to download files from the manifest 38.

The manifest 38 may include all components, including dlls, component add-ins, Extensible Markup Language (XML), schema files and all associated XML files

required by a software application for operating properly or required for improving, or adding, functionality to the software application 100. According to an embodiment of the present invention, the manifest 38 may also include information helpful to the end user, or to the end user's computer, for installing the downloaded components. For example, the manifest 38 may include information specifying the type for a given dynamic-link library (dll) so that the client-side computer 20 can properly register the dll including any need for specific registry keys for properly registering the dll.

A schema may be attached to the manifest 38 to define permissible data content, data type and data structure of the manifest as a file and for associating the manifest with files, documents and applications that may call the manifest 38 for obtaining a download of required components. For purposes of example only and not for limitation of the invention described herein, the following is an example XML-based schema that may be attached to the manifest 38 for associating the manifest with files, documents and applications that may call the manifest 38 for obtaining a download of required components.

Sample Manifest Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:schemas-microsoft-com:smartdocuments:manifest"
    targetNamespace="urn:schemas-microsoft-
com:smartdocuments:manifest" elementFormDefault="qualified">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
      The schema for office smart document, transform and schema manifests.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="manifest" type="manifestType" />
  <xsd:complexType name="manifestType" mixed="true">
    <xsd:sequence>
      <xsd:element name="version" type="xsd:string" />
      <xsd:element name="location" type="xsd:string" />
```

```
                    <xsd:element name="updateFrequency"
type="zeroAndPositiveInteger" minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="uri" type="xsd:string" />
                    <xsd:element name="manifestURL" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                    <xsd:element name="solution" type="solutionDefn"
minOccurs="1" maxOccurs="unbounded" />
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="solutionDefn">
                <xsd:sequence>
                    <xsd:element name="solutionID" type="xsd:string" />
                    <xsd:element name="type" type="solutionType" />
                    <xsd:element name="alias">
                      <xsd:complexType>
                        <xsd:simpleContent>
                          <xsd:extension base="xsd:string">
                          <xsd:attribute name="lcid" type="xsd:string"
use="required"/>
                          </xsd:extension>
                        </xsd:simpleContent>
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="solutionSpecific" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                    <xsd:element name="file" type="fileType" minOccurs="1"
maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:simpleType name="solutionType">
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="schema" />
                    <xsd:enumeration value="transform" />
                    <xsd:enumeration value="smartDocument" />
```

```
            </xsd:restriction>
        </xsd:simpleType>
        <xsd:complexType name="fileType">
            <xsd:all>
                <xsd:element name="runFromServer" type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="type" type="typeDefn" />
                <xsd:element name="application" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="version" type="xsd:string" />
                <xsd:element name="context" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="filePath" type="xsd:string" />
                <xsd:element name="installPath" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="register" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="CLSID" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="progID" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="regsvr32" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="regasm" type="xsd:string" minOccurs="0" maxOccurs="1" />
                <xsd:element name="registry" type="registryType" minOccurs="0" maxOccurs="1"/>
            </xsd:all>
        </xsd:complexType>
        <xsd:simpleType name="typeDefn">
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="template" />
                <xsd:enumeration value="smartTagList" />
                <xsd:enumeration value="solutionList" />
```

```
                    <xsd:enumeration value="schema" />
                    <xsd:enumeration value="transform" />
                    <xsd:enumeration value="actionHandler" />
                    <xsd:enumeration value="solutionActionHandler" />
                    <xsd:enumeration value="recognizer" />
                    <xsd:enumeration value="solutionRecognizer" />
                    <xsd:enumeration value="solutionList" />
                    <xsd:enumeration value="COMAddIn" />
                    <xsd:enumeration value="assembly" />
                    <xsd:enumeration value="XMLFile" />
                    <xsd:enumeration value="installPackage" />
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:complexType name="registryType">
                <xsd:all>
                    <xsd:element name="registryKey" type="registryKeyType"
minOccurs="1" />
                </xsd:all>
            </xsd:complexType>
            <xsd:complexType name="registryKeyType">
                <xsd:all>
                    <xsd:element name="keyName" type="xsd:string" />
                    <xsd:element name="keyValue" type="keyValueType" />
                </xsd:all>
            </xsd:complexType>
            <xsd:complexType name="keyValueType">
                <xsd:all>
                    <xsd:element name="valueName" type="xsd:string" />
                    <xsd:element name="valueType" type="registryValueType" />
                    <xsd:element name="value" type="xsd:string" />
                </xsd:all>
            </xsd:complexType>
            <xsd:simpleType name="registryValueType">
                <xsd:restriction base="xsd:string">
```

```
                    <xsd:enumeration value="REG_SZ" />
                    <xsd:enumeration value="REG_EXPAND_SZ" />
                    <xsd:enumeration value="REG_DWORD" />
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="zeroAndPositiveInteger">
                <xsd:restriction base="xsd:integer">
                    <xsd:minInclusive value="0"/>
                </xsd:restriction>
            </xsd:simpleType>
            </xsd:schema>
```

If the user has opened a software application 100 that requires the functionality of various components to transform the user's document 110 into a "smart" document, as described below, the mechanism of the present invention may be used to download all components required for that operation. In that case, an XML schema that describes the type(s) of files and data structures that together make up a solution to transform the user's document 110 into a "smart" document, as well as information about registering those components and installing them on the user's computer 20 may be stored on the manifest 38.

For example, if the user receives a document, such as a template, for the preparation of a resumé that "points" to a solution for providing helpful information, and actions, associated with completing the text and data fields of the template, the template or document received by the user may point to a remotely stored manifest 38 so that the user may download all files necessary for fully implementing the solution referred to by the document received by the user. For a detailed description of a method and system for creating, implementing and using "smart" documents such as the document 110, illustrated in Fig. 3, see U.S. Patent Application entitled "*Providing Contextually Sensitive Actions and Help Content In Computer-Generated Documents,*" Serial No. _____, filed _____, which is incorporated herein by this reference as if fully set out herein.

Once the manifest 38 is created, and all of the necessary files are listed to implement a given solution, or to correct or improve the performance of a given application, a reference to the solution or updates may be made in the document, such as a word processing document, or spreadsheet document utilized by the user on the client-side computer 20. The manifest of files and solutions may be referred to by a solution or manifest ID to allow the user's client-side application and documents to point to and obtain information from, the remote manifest. The solution ID associated with software components pointed to by the document or application are stored in a solution property 115 attached to the document 110.

The location of the manifest 38, including required components and desired solutions, is referred to according to the solution location 118. The solution location 118 may include the URL of the manifest 38 on the remote web server 49. If the user only has the document, as in the case where the user has received the document from a friend via electronic mail, the application 100 may call the web server 49 via the solution location 118, and by utilizing the solution ID from the solution property 115, the application may obtain the manifest 38 to determine what components must be downloaded, installed, registered, and implemented to provide the user with required or desired functionality. Now, the user has the document 110 and a set of installed "ready to run" files and other components that the software document 110 needs to enable the proper operation of the software application 100 or to enable the application 100 to transform the document 110 into a "smart" document. Advantageously, the requirement to receive an installation package from the software developer containing software repairs or "patches" or containing necessary functionality to improve the performance of original functionality of the application is avoided.

Fig. 2 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules

include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, cell phones, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 2, an exemplary system for implementing the invention includes a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples the system memory to the processing unit 21. The system memory 22 includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27, a magnetic disk drive 28, e.g., to read from or write to a removable disk 29, and an optical disk drive 30, e.g., for reading a CD-ROM disk 31 or to read from or write to other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage for the personal computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs 100, a word processor program module 37 (or other type of program module), program data, such as the manifest 38, and other program modules (not shown).

A user may enter commands and information into the personal computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Fig. 2. The logical connections depicted in Fig. 2 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the LAN 51 through a network interface 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the WAN 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote

memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Fig. 3 is a computer screen display of a software application graphical user interface through which is displayed a document and associated contextually sensitive actions and help content according to an embodiment of the present invention. The document 110 illustrated in Fig. 3 represents a "smart" document that may point to functionality that may be downloaded from the manifest 38. The document 110 illustrated in Fig. 3 is described for purposes of example only and is not limiting of the invention described herein. The word processing application 100 provides typical functionality associated with a word processor accessible via drop down menus such as, File, Edit, View, Insert, Format, etc. The document 110 is displayed in the work area of the application 100, and a document actions pane 135 is illustrated to the right of the document 110.

When the user places her computer cursor within a particular section of the document 110, for example the "objectives" section 125 illustrated in Fig. 3, the user is provided with actions and help content in the document actions pane 135. For example, if the user places her computer cursor in the "objectives" section 125, the user may be provided with "Objective Writing Tips" 155 shown in the document actions pane 135. Selection of the "Objective Writing Tips" 155, as illustrated in Fig. 3, causes a display of "Objective Writing Tips" text 160 that provide the user with helpful information as to how to complete the "objectives" section the performance review document 110, illustrated in Fig. 3. If the user moves the cursor to a different section of the document, for example the personal information section 120, information provided in the document actions pane 135 will be provided to give the user assistance with the completion of the personal information section 120.

In addition to helpful information to assist the user, a variety of document actions 145 are provided. For example, the "Submit Review" action may allow the user to submit the completed document 110 to her supervisor or to her employee after completion of the document. The "Open Last Review" action may allow the user to

open the last performance review so that she may determine whether she completed her objectives as set forth in the last review. If the document in use by the user is some other type of document, for example a resumé document, helpful information in the document actions pane might include information on preparing the "education" section, the "experience" section, and/or the "personal information" section.

According to the present invention, and as described in detail below, when a user focuses on a particular portion of the document 110, such as the "objectives" section of the performance review 110 illustrated in Fig. 3, a solution property 115 points the document to the "objectives" section help solution illustrated in the document actions pane 135. The solution location 118 provides the document 110 and the application 100 with the location of the components, dlls, or XML files necessary for implementing that solution. As should be understood, exemplary components may include components for drawing the document actions pane 135, components for displaying the information associated with the particular context, in this case the "objectives" section, and components for executing document actions such as "Submit Review" action 145. If the components needed by the application 100 and document 110 to provide the "smart" functionality described above have not been downloaded to the user's computer 20, then these components may be downloaded to the computer 20 in accordance with the present invention as described below.

Fig. 4 is a flow chart illustrating a method for downloading components required for operating and for providing functionality to a client-side software application. As described above, the embodiments of the present invention provide for downloading all components, files, sub-routines, Extensible Markup Language files, and file implementation information necessary for the operation of the software application 100 on the user's computer 20. The embodiments of the present invention also provide for downloading all necessary components, files, XML schema, and associated XML files required by an application 100 for transforming a document 110 into a "smart" document.

The method 400 for downloading required components and files to the user's computer 20, begins at start step 405, and proceeds to step 410 where the

application 100 is booted by the user, and the document 110 is opened for editing by the user. According to an embodiment of the present invention, the document 110 opened by the user may be any document that points to required components and files for operation of the document 110 via the application 100, or the document may be a "smart" document that points to a solution that may be downloaded to provide enhanced functionality to the document, as described above with reference to Fig. 3.

At step 415, the application 100 determines whether the document 110 points to or refers to components, sub-routines, or files that are necessary for the operation of the application 100, or whether the document 110 points to a solution that may be downloaded to the user's computer 20 to provide additional functionality to the application 100 for editing the document 110. According to an embodiment of the present invention, a schema is attached to a document 110 that defines permissible data content, data type and data structure from the document. The document is structured to associate the document with the schema. A solution comprising components and files needed to provide functionality to the application or to transform the application into a "smart" document is associated with the document structure.

According to another embodiment of the invention, an Extensible Markup Language (XML) schema is attached to the document to define permissible data content, data types and data structures for the document. Particular portions of the document are annotated with XML structure associated with the schema. A solution comprising a plurality of software components is associated with XML elements making up the XML structure. The document is enabled to call the solution to obtain functionality provided by the plurality of software components assembled on the manifest 38 upon initiation of editing the document within an XML element associated with a particular solution.

If the document 110 points to various components and files necessary for the operation of the application 100 and for editing the document 110, or if the document points to a solution that may be downloaded to transform the document into a "smart" document, the method proceeds to step 420, and the application 100 checks the schema library 105 resident on the user's computer 20 for the presence of the necessary

components or files. If the document includes solution properties pointing to more than one solution, the user of the document may be prompted to select one of the solutions for downloading from the remote server.

Referring back to step 415, if the document does not refer to a solution for transforming the document into a "smart" document, the method may proceed to step 425, where a determination is made as to whether a schema has been attached to the document. If the document has an attached schema, but no reference to a particular solution, the method proceeds to step 420, and the schema library 105 is checked for the presence of components associated with the schema attached to the document 110. If a determination is made that the document does not refer to any required components or files, or that the document does not have an attached schema, the method proceeds to step 460, and the document is opened in accordance with normal document opening procedures according to the document application 100.

Referring back to step 420, at the schema library 105, a determination is made as to whether the schema library 105 includes files associated with the application 100 and the document 110. A determination as to whether a solution is present in the schema library 105 that is referred to in the document 110 is made by looking for component or solution IDs referred to in the solution properties 115 in the document 110.

At step 430, a determination is made as to whether a download of components, files, sub-routines, or information related thereto should be downloaded to the user's computer 20 to update existing components or to add additional functionality to transform the document 110 into a "smart" document. In order to make the determination as to whether components should be updated, or as to whether additional functionality should be downloaded from the manifest 38, the user may be provided with a prompt, such as the dialog box 600, illustrated in Fig. 6. The dialog box may inform the user that components necessary for the proper operation of the application 100 may have been updated or repaired recently, or that the document she is currently editing will work better if the user downloads components from a given web server site. As shown in the dialog box 600, the user may be provided with a variety of options including acceptance of a download of components, acceptance of

automatically downloading future updates, or the user may decline the downloading of updates or components.

A registry key may be written into the computer's registry so that when the application boots, a query can be sent to the manifest 38 via the remote server 49 to determine whether new functionality has been added to the manifest 38 that is associated with the application 100 or with solutions being used by the application 100 to transform the application into a "smart" document. Alternatively, the manifest 38 may include a timing mechanism that contacts the user after a set amount of elapsed time to notify the user to check for updates to files or functionality contained on the manifest 38. The amount of time between checks for updates to the manifest 38 may be set by the creator of the manifest 38 or by the user of the application 100, as described below. Alternatively, the application 100 can be programmed to call or "ping" the server 49 on some regular frequency to determine whether any software component updates or additions are available. Alternatively, the user may request software component updates to be downloaded from the manifest 38 upon user demand, or the user may choose to have updates downloaded on some regular frequency, such as application boot-up, or daily, weekly, etc. According to an embodiment, the user may be provided with a user interface for choosing the frequency of download from the manifest 38.

At step 435, if the user decides to download updates to existing files or components or new functionality to transform the document 110 into a "smart" document, the user requests the download of the suggested files or functionality, and the application 100 connects to the remote server 49 to request the required files or updates from the manifest 38. In order to connect the application 100 to the remote server 49 to obtain downloaded information from the manifest 38, the application 100 may launch an instance of an Internet browser capable of connecting the user's computer 20 to the remote server 49. Alternatively, the functionality may be integrated into the application 100 for connecting to the remote server 49 in the same manner as an Internet browser would connect to the remote server 49 to download information directly to the application 100 via the user's computer 20. If the user's computer 20 is not adapted for

on-line communication, a prompt such as the dialog box 700 illustrated in Fig. 7 may be provided to the user to inform the user that the document may not work properly if the user is off-line.

As should be understood, the call made from the application 100 to the manifest 38 located on the remote server 49 may be made based on a uniform resource locator (URL) designated for obtaining updates to existing files or new functionality and attached to the document at the solution location 118, illustrated in Fig. 1. Alternatively, the user may have a set of URLs from which to obtain updates to software components or additional functionality, and the user may enter one or more of those URLs in response to a prompt informing the user that updating the software components provisioned for her application 100 may be helpful. Alternatively, the user may decide without being prompted to cause the application 100 to call the manifest 38 to obtain software component updates and additions. According to this embodiment, a function tool may be provided in the application 100 that allows the user to enter the URL of a manifest 38 from which the user desires to obtain software component updates or functionality additions.

Once the user has requested the download of software component updates or functionality additions for use by the application 100, the method proceeds to step 440 and a number of security checks are performed to ensure that the user receives the downloaded files and updates from a trusted source. If the document refers to a manifest 38 via a server site that is on the user's list of trusted sites, or if the manifest is requested from a site on the user's trusted intranet system, then the download of the requested or accepted files or updates may be performed. If the URL at which the manifest 38 is located is not a trusted site or in the user's trusted Intranet system, the user may be notified with a prompt such as the dialog box 800, illustrated in Fig. 8, to notify the user that the site being called is not on the user's list of trusted sites or is not coming via the user's trusted intranet system. The user may then choose to either accept a download of information from the manifest 38 or decline the download in response to the prompt. If the document refers to a manifest 38 that is on a trusted site or that is available via a trusted Intranet system, but where the site containing the

manifest 38 is not available, the user may be notified with a prompt such as the dialog box 900 illustrated in Fig. 9 that the site is currently unavailable and that the download of information is likewise not available.

An additional security check that may be performed after the download, but prior to installing components, includes preparing a checksum value of the information provided by the manifest 38 and comparing that to a checksum value prepared for the downloaded information at the user's computer 20. If the two values do not match, the installation to the user's computer 20 is prevented because the variation in the checksum values may be an indication that unauthorized tampering with the contents of the downloaded files occurred in transit to the user's computer 20.

In addition to the foregoing security checks, digital signatures may also be utilized to check the security of files downloaded from the manifest 38. That is, files contained on the manifest 38 may be enhanced with a digital signature of the creator or administrator of those files that may be checked at the remote server 49 or at the user's computer 20 against a list of trusted digital signatures prior to downloading and/or installing information from those files. If any of the foregoing security checks fail to ensure the validity and security of the files contained in the manifest 38, the method proceeds to step 460, and document opening is completed without downloading any additional components or file updates. That is, the document opens and operates without the benefit of new functionality or updates to existing functionality that may be available from the manifest 38. Alternatively, the user may override the security system and accept the downloaded information even if the security check indicates that the information is not coming from a trusted site if the user is otherwise confident that the downloaded information may be trusted.

Referring back to step 445, if the security checks pass, the method proceeds to step 450 where a validation of the contents of the manifest 38 is performed. Before downloading components or information from the manifest to the user's computer 20, a comparison is made between the components and information contained in the manifest 38 (pointed to by the document 110) and the components and information already present in the schema library 105. If the components and information on the manifest

38 does not differ from the components and information already available to the application via the schema library 105, then the download from the manifest 38 may be avoided.

The validation of the manifest 38 includes a determination of which components or information regarding those components are pointed to by the document 110 via the application 100. That is, a solution referred to by the document 110 may require additional functionality, and the solution may point to particular components contained on the manifest 38, but not point to all components contained on the manifest 38. Accordingly, at step 450, a determination is made as to the number of components and the content of information related to those components that should be downloaded to the user's computer 20 for integration with the application 100.

At step 455, the manifest 38 is processed. Processing the manifest 38 includes actually downloading files, components, subroutines, and related information pointed to by the document 110 or application 100. Once the required files are downloaded to the user's computer 20, those files are installed and registered with the operating system of the computer 20. As should be understood by those skilled in the art, any required registry keys or set-up actions necessary to ensure that the downloaded files properly correspond with the application 100 to provide the required functionality are performed at step 455. After all required files are downloaded from the manifest 38, opening the document 110 is completed with the additional or updated functionality provided by the downloaded file updates or file additions. The method ends at step 490.

Once the document 110 is opened after the download of component updates or additions, the user of the document now has all available functionality downloaded onto the user's computer 20. If the document is a "smart" document as illustrated with reference to Fig. 3, the document will provide the user with all the normal functionality required for operating the document, and additionally, the user will have useful help content and document actions based on the editing context within the document. For example, if the user has opened a resume template, the user may receive help content and document actions based on the position of the user's computer cursor in a section of the document. In addition to pointing to the solution ID for obtaining the solution for

use by the document, the document will point to the solution location from where the software components or software component upgrades may be downloaded to upgrade or add to the functionality in use by the document. Accordingly, with the present invention, the user may readily obtain downloaded updates and additions to the functionality available for use with the document being edited.

Moreover, if the user would like to send the document to a second user, there is no need to send the second user an installation package containing all the software components necessary to give the document the enhanced or "smart" functionality. Once the second user opens the document, the second user will be notified that the document will work more efficiently if the user downloads software components from the manifest 38, as discussed with reference to Fig. 4. If the downloaded information is accepted, the document will then have all the available functionality on the second user's computer 20 that it had on the computer of the first user.

As described above, with reference to Fig. 1, a manifest may be utilized for obtaining software components for enhancing a software application or for providing a solution to a document to make the document more intelligent or "smart." According to an alternate embodiment, a manifest may be utilized for dynamically configuring a document solution based on information obtained about the user of the document. Fig. 10 is a block diagram illustrating a system architecture for use in conjunction with an embodiment of the present invention showing interaction between a user's computer and a remotely maintained manifest and manifest collection. Referring to Fig. 10, according to this embodiment of the present invention, when a user obtains or creates a document that references or points to a document solution requiring that the solution or the components of the solution be obtained from the manifest 1038, a request is sent from the user's document via the user's computer 20 to the manifest 1038 via the server 49. According to this embodiment of the present invention, the manifest 1038 resides in an active server page 1000 having computer executable instructions sufficient for dynamically configuring a desired or required document solution based on information obtained about the user of the document solution.

When the request is forwarded from the user's document to the manifest 1038 to obtain the desired or required document solution, identification of the user is also sent to the manifest 1038. At the manifest 1038, operating in the active server page 1000, the manifest 1038 passes a query to the user database 1050 to obtain profile information for the user based on the identity of the user. According to embodiments of the present invention, the user database may be in the form of an active directory database. In response to the database query to the user database 1050 user profile information such as the user's job title, the user's work location, the user's educational background, or any other information available concerning the user is passed to the manifest 1038.

Based on information obtained from the user database 1050, the manifest 1030 obtains solution components 1045 for dynamically configuring the solution requested by the user for the user's document. For example, consider that a first user opens an employee review form which points to a document solution that may be obtained from the manifest 1038. If the user profile returned from the user database 1050 indicates that the user is a human resources manager, the manifest 1038 may obtain solution components 1045 for dynamically configuring and tailoring the document solution for use by a human resources manager. For example, one solution component may provide help functionality for assisting the human resources manager in completing fields of the employee review form associated with human resources issues. On the other hand, if a second user utilizing the same employee review form requests a document solution from the manifest 1038, and the user profile for the second user indicates that the second user is a training manager, a different document solution may be configured by the manifest 1038. For example, solution components 1045 may be assembled by the manifest 1038 such that the resultant document solution adds additional fields to the employee review form for completion by a training manager or which provide help functionality to the training manager for rating the training performance of a given employee. Accordingly, instead of the two different users receiving the same document solution, each user receives a document solution that is tailored to their needs based on profile information provided to the manifest 1038 about the individual users.

In cases where a user receives a document, such as a template associated with one or more document solutions, by electronic mail distribution, copying from a shared file, or some other location apart from receiving the document and document solution from the manifest 1038, some software applications operate security systems causing those applications not to trust such a document if the document is not installed by the user in an appropriate location. For example, some software applications have a particular memory location for storing templates. If the user imports a template document from some external source not via the manifest 1038, as described above, the template document and the document solution may not operate properly if the code of the software application cannot locate the template and the associated solution, particularly if the manifest processing code of the software application expects the template document to reside in a particular location.

According to the embodiments of the present invention, when a solution designer builds a document solution, such as a template document, the solution designer adds a property to the document that identifies the document as being part of or associated with a document solution. When a user subsequently opens the components of the document solution or a document based on the document solution, the software application detects the presence of the property added to the document by the solution designer. The software application then checks to see if a solution is referenced by the document. If a solution is referenced by the document, a solution directory is checked to see if this document and solution is presently in the solution directory of the software application. If the same document solution, such as a template, is already present, a comparison is performed of the two document solutions, and the older version of the document solution is replaced by the newer version of the document solution in order to allow the user to benefit from updates to the document solution. Now, the document solution obtained or received by the user is saved to a location expected by the software application and manifest processing code associated with the software application for processing instructions to and from the manifest 1038 and for processing information received from the manifest 1038 such as the document solution.

Referring still to Fig. 10, a manifest collection 1040 containing one or more namespace/solution pairs 1055 is illustrated. According to embodiments of the present invention, many entities such as large corporations, large educational institutions, large civic or government organizations, as well as individual users may maintain many documents associated with document solutions, and moreover, document solutions for the numerous documents maintained by such entities and users may be updated frequently. For example, a large shipping organization may have tens or hundreds of shipping and purchase order forms annotated with document solutions, such as Extensible Markup Language (XML) based document solutions each pointing to one or more XML namespaces (schemas) for defining the operational and structural rules associated with markup structure applied to the document in association with the document solutions. If an organization attempts to associate each of the numerous document solutions for all the available namespaces potentially referenced by individual documents onto each user's computer 20, excessive memory space will be required, inefficiencies will be increased, and frequent updating required will be cumbersome and time consuming.

According to embodiments of the invention, the manifest collection 1040 contains a listing of namespace/solution pairs 1055 for directing a document to a document solution maintained at one or more manifests 1038. The manifest collection 1040 serves as a central repository for maintaining a listing of solutions for all the different namespaces referenced by documents maintained an operated by the organization. On each user's computer 20, a registry key is designated for pointing the user's computer 20 to a uniform resource locator (URL) associated with the manifest collection. If the user opens a document referencing a document solution associated with one or more namespaces, the document may point to the URL of the manifest collection via the registry key set on the user's computer 20 and pass the name of a given document namespace to the manifest collection 1040.

At the manifest collection 1040, the namespace is compared against namespace/solution pairs to determine if a matching namespace is in the manifest collection. If the namespace passed from the user's document matches a particular

namespace/solution pair 1055, the solution identification and location for the namespace/solution pair are utilized to locate a particular document solution at the manifest 1038 for returning to the user for application to the user's document. According to an embodiment of the present invention, if a namespace from the user's document matches a namespace/solution pair 1055, an XML file is returned which describes a mapping of namespace uniform resource identifiers (URI) to a location of a manifest 1038 for obtaining desired or required document solutions. Accordingly, the organization operating the manifest collection 1040 may make new or updated solutions available to all its users without the need for updating each user's software applications, documents, or computer. For example, if a given document solution associated with a given document namespace requires updating, the organization may obtain the updated document solution referenced by a given namespace/solution pair 1055. When a given user's document subsequently references that namespace/solution pair, the user will obtain the updated document solution without additional user input.

Fig. 11 is a flowchart illustrating a method for obtaining a dynamically configured document solution for use in conjunction with a user document according to embodiments of the present invention. Referring to Fig. 11, an example operation of the manifest 1038 and manifest collection 1040 described above with reference to Fig. 10 will be described. The method 1100 begins at start block 1105 and proceeds to block 1110 where a user opens a document, for example, an employee review template, at the user's computer 20. At block 1115, a determination is made as to whether the document contains a solution ID and namespace associated with a document solution. If so, a determination is made as to whether a solution matching the solution ID and namespace is available in a local schema library 105 as described above with reference to Figs. 1 and 4.

If the required solutions may be found in the user's local schema library 105, the method proceeds as described with reference to Fig. 4, block 420. If the required solution is not available from the user's local schema library 105, the method proceeds to block 1125, and a determination is made as to whether the user's document points to a manifest 1038, as described above with reference to Fig. 10. If so, the method

proceeds to block 1145, and a call is made to the manifest 1038 along with identification information for the user for obtaining a dynamically configured document solution tailored for this particular user.

At block 1155, the solution components obtained by the manifest 1038 operating as an active server page 1000 are passed to the host application operating the user's document at the user's computer 20. At block 1160, the user's document is opened with the solution applied to the document.

Referring back to block 1125, if the document refers to a document solution or namespace which is not located at the user's local schema library 105, and the document does not point to a manifest, the method proceeds to block 1130 and the manifest collection 1040 is called using the URL set for the user's computer 20. At block 1135, a determination is made as to whether the namespace associated with the user's document matches a namespace/solution pair 1055 contained in the manifest collection 1040. If no match is obtained, the method proceeds to block 1140 and the document is opened without the benefit of a document solution. On the other hand, if a matching namespace/solution pair is located in the manifest collection 1040, the solution ID and location information is utilized for calling a particular manifest 1038 where the solution components may be obtained. The method then proceeds to block 1145 for obtaining the document solution, as described above. Alternatively, if the document does not point to a manifest nor refer to a namespace, a processing instruction may be used to point the document to the manifest 1038 or to the manifest collection as discussed below. As is appreciated by those skilled in the art, a processing instruction allows a user to cause the document to reference a manifest or the manifest collection where the document does not point to the manifest or where the document does not refer to a namespace.

Referring back to block 1115, as should be appreciated, even if it is determined that a referenced document solution or namespace is located at the user's local schema library 105, according to an alternate embodiment, a call may nonetheless be made to one or both of the manifest 1038 or manifest collection 1040 for determining whether any updates to the document solution contained in the user's local schema library 105

have been made. If updates have been made, the updated solutions may be obtained, as described herein for updating the solution contained in the user's local schema library 105.

It will be apparent to those skilled in the art that various modifications or variations may be made in the present invention without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein.